

Increasing Fault Resiliency in a Message-Passing Environment

CS 591: Extreme Scale Distributed Systems

Rolf Riesen

rolf@sandia.gov

Sandia National Laboratories

April 21, 2010

Motivation
Design
Evaluation
Simulation
Validation
Results
Summary

Motivation

Design

Evaluation

Simulation

Validation

Results

Summary

Motivation

Daly

MTBF

Tradeoff

Design

Evaluation

Simulation

Validation

Results

Summary

Motivation

Motivation

Daly

MTBF

Tradeoff

Design

Evaluation

Simulation

Validation

Results

Summary

- Checkpoint/Restart is common way to deal with faults
- What can we do to increase checkpoint interval?
- Explore redundant computation for MPI applications
 - ◆ Can it be done at user level?
 - ◆ What are requirements for RAS system and runtime?
 - ◆ What is the overhead
 - ◆ Cost versus benefit?
- Write *rMPI* library at MPI profiling layer to learn what the issues are

Motivation
Daly
MTBF
Tradeoff

Design

Evaluation

Simulation

Validation

Results

Summary

$T_w(\tau)$	Wall clock time to complete application run
Θ	Application MTBF
R	Restart time
τ	Checkpoint interval
δ	Checkpoint time
T_s	Solve time

$$T_w(\tau) = \Theta e^{\frac{R}{\Theta}} (e^{\frac{\tau+\delta}{\Theta}} - 1) \frac{T_s}{\tau} \quad \text{for } \delta \ll T_s \quad (1)$$

Motivation

Daly

MTBF

Tradeoff

Design

Evaluation

Simulation

Validation

Results

Summary

System MTBF Θ_{sys} depends on node MTBF

$$\Theta_{\text{sys}} = \frac{1}{\frac{1}{\Theta_1} + \frac{1}{\Theta_2} + \dots + \frac{1}{\Theta_n}} = \frac{1}{n \frac{1}{\Theta}} = \frac{\Theta_{\text{node}}}{n} \quad (2)$$

Motivation

Daly

MTBF

Tradeoff

Design

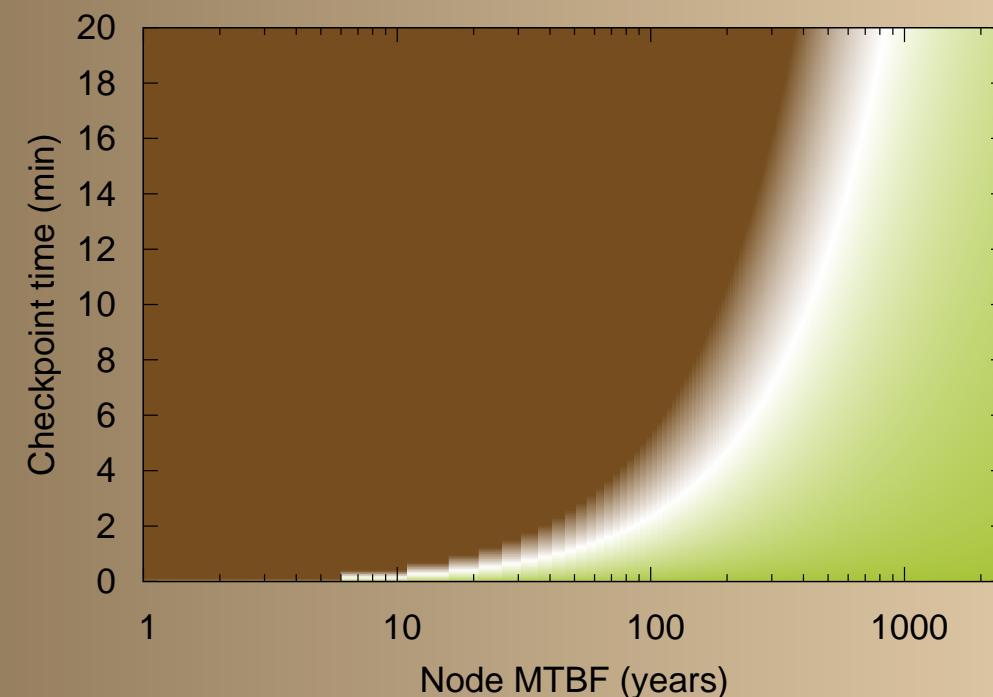
Evaluation

Simulation

Validation

Results

Summary



- When does using twice the nodes pay off?

Motivation

Design

Redundancy

Mirror

Msg. order

Other

Parallel

Status

Evaluation

Simulation

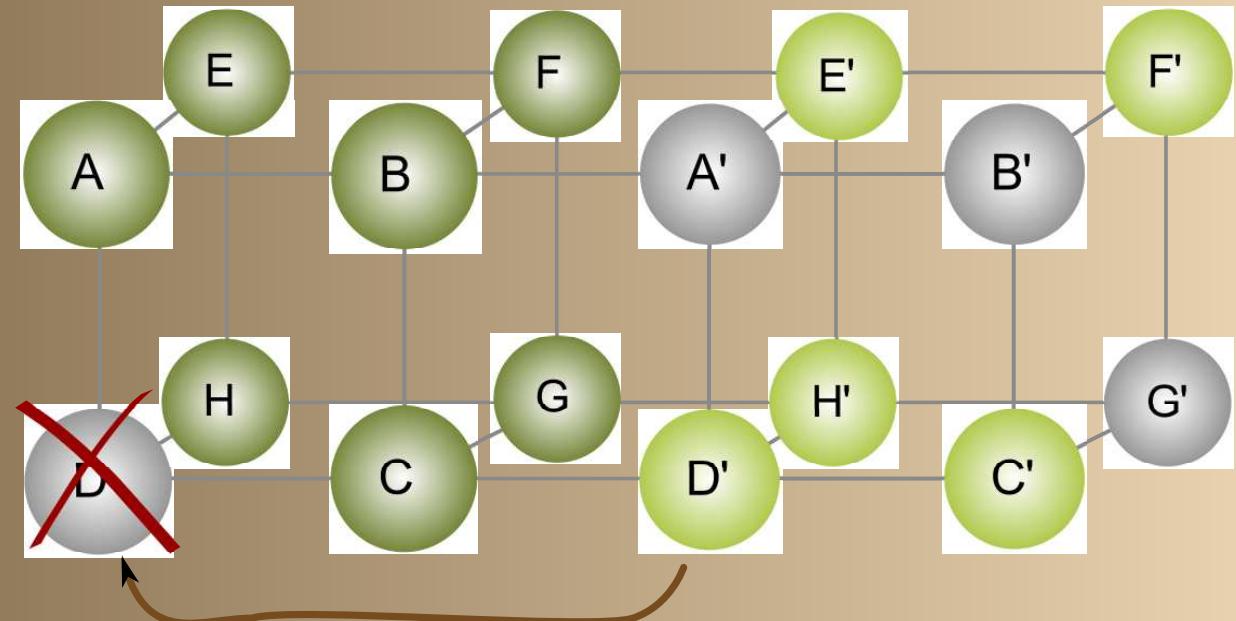
Validation

Results

Summary

Design

Motivation
Design
Redundancy
Mirror
Msg. order
Other
Parallel
Status
Evaluation
Simulation
Validation
Results
Summary

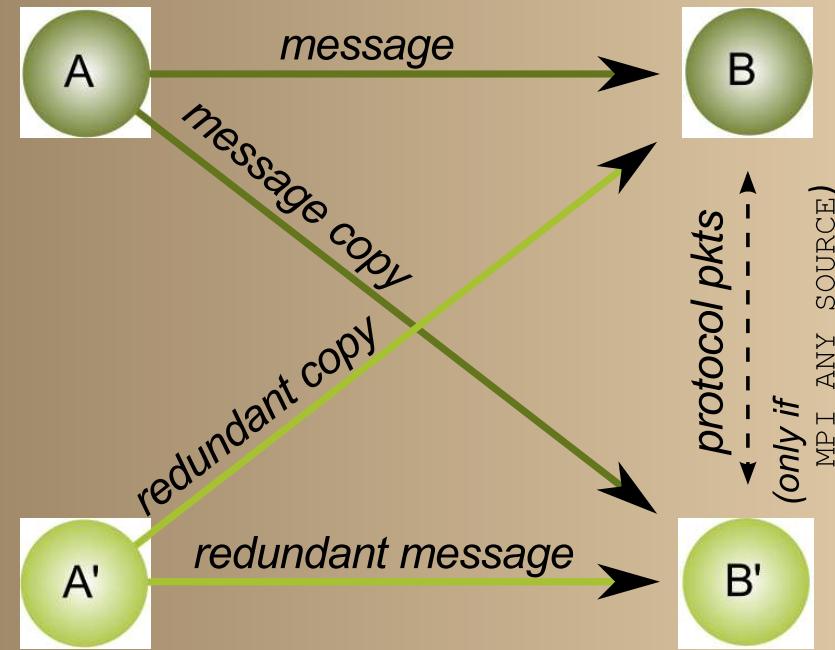


- Active and redundant node do same computation
- One node continues when the other fails
- `MPI_Comm_rank()` returns same value on both nodes
- Not each node needs to have a redundant partner

Basic Operation of Mirror Protocol

Motivation
Design
Redundancy
Mirror
Msg. order
Other
Parallel
Status

Evaluation
Simulation
Validation
Results
Summary



- Each message gets sent twice (4, if we count redundant nodes)
- Msg and redundant copy received into same buffer
- Redundant msg have unused tag bit set
- Protocol needed to coordinate receives and other MPI ops

Motivation

Design

Redundancy

Mirror

Msg. order

Other

Parallel

Status

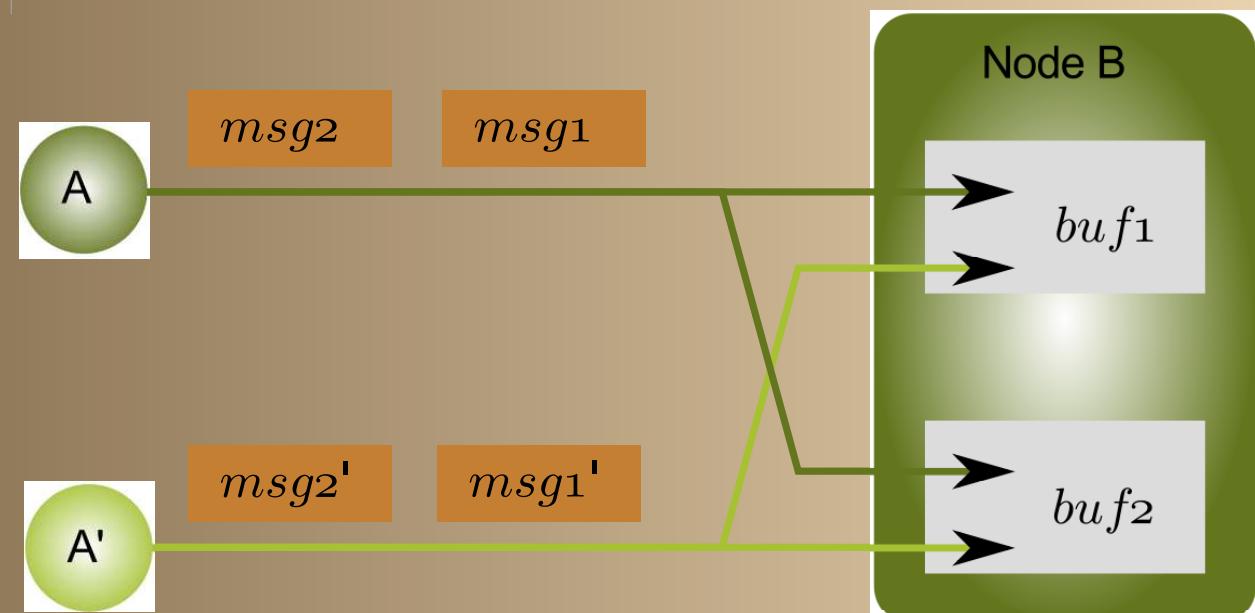
Evaluation

Simulation

Validation

Results

Summary



- Active and redundant node must receive msg in same order
- MPI_ANY_SOURCE and MPI_ANY_TAG are problematic
- Redundant node maintains queue of posted receives if MPI_ANY_SOURCE
- Coordinate with active node to post specific receive

[Motivation](#)[Design](#)[Redundancy](#)[Mirror](#)[Msg. order](#)[Other](#)[Parallel](#)[Status](#)[Evaluation](#)[Simulation](#)[Validation](#)[Results](#)[Summary](#)

- Redundant node must return same info for probe, test, and time functions
- Active node does operation and sends result to redundant node
- Collectives use redundant point-to-point
- *rMPI* re-implements almost all of MPI
 - ◆ *rMPI* uses MPICH (mostly) as a transport layer

Motivation

Design

Redundancy

Mirror

Msg. order

Other

Parallel

Status

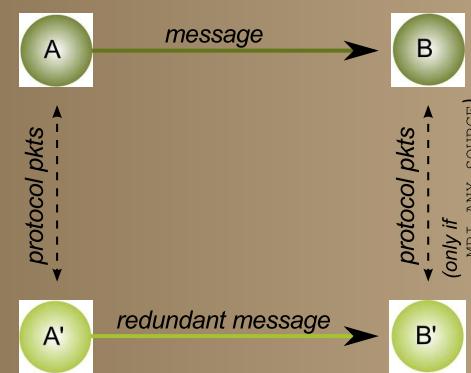
Evaluation

Simulation

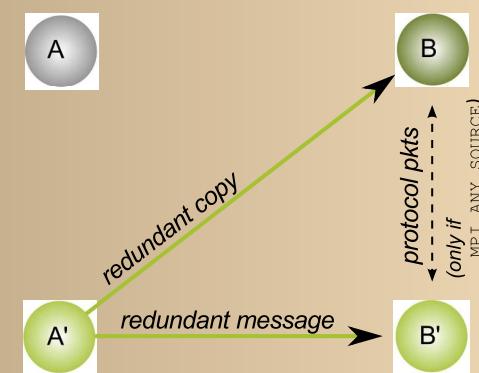
Validation

Results

Summary



(a) Protocol without failed nodes



(b) Protocol with a failed sender

- Senders need to coordinate: one, and only one, msg to each receiver
- Fewer large messages
- Degenerates to mirror protocol when nodes fail

Motivation

Design

Redundancy

Mirror

Msg. order

Other

Parallel

Status

Evaluation

Simulation

Validation

Results

Summary

- Can't do MPI_ANY_TAG and MPI_ANY_SOURCE simultaneously
- Most major functions of MPI-2 implemented
- Plan to open source it
- Few RAS features needed:
 - ◆ Notification of node availability
 - ◆ No Byzantine behavior (error correction protocol on network)
 - ◆ Messages to/from dead nodes must be consumed (no dead- or life-lock)
 - ◆ Parallel requires more info from RAS system: which nodes are alive?

Motivation

Design

Evaluation

Bandwidth

Latency

LAMMPS

SAGE

CTH

HPCCG

Sys MTBF

Birthday

Trade-off

Simulation

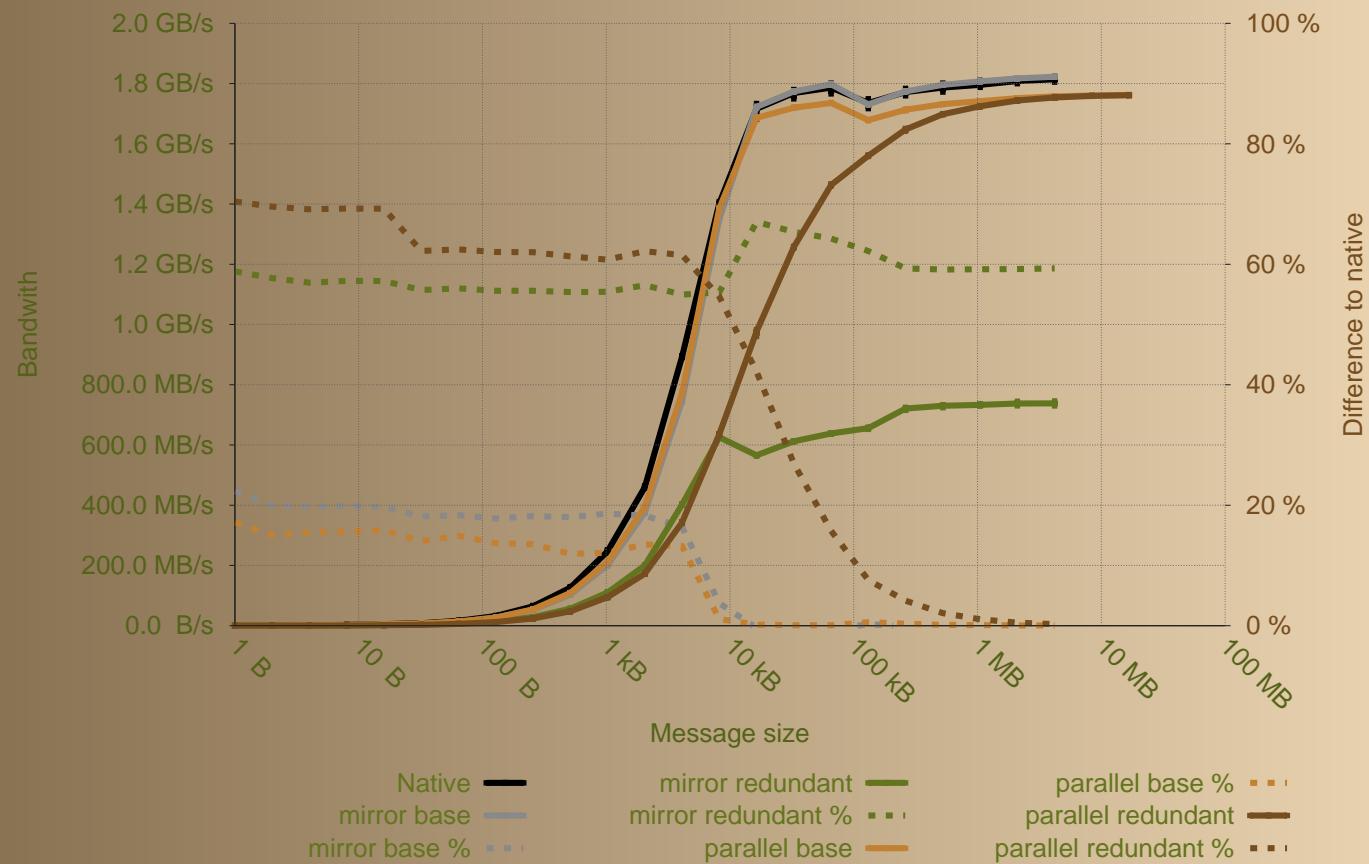
Validation

Results

Summary

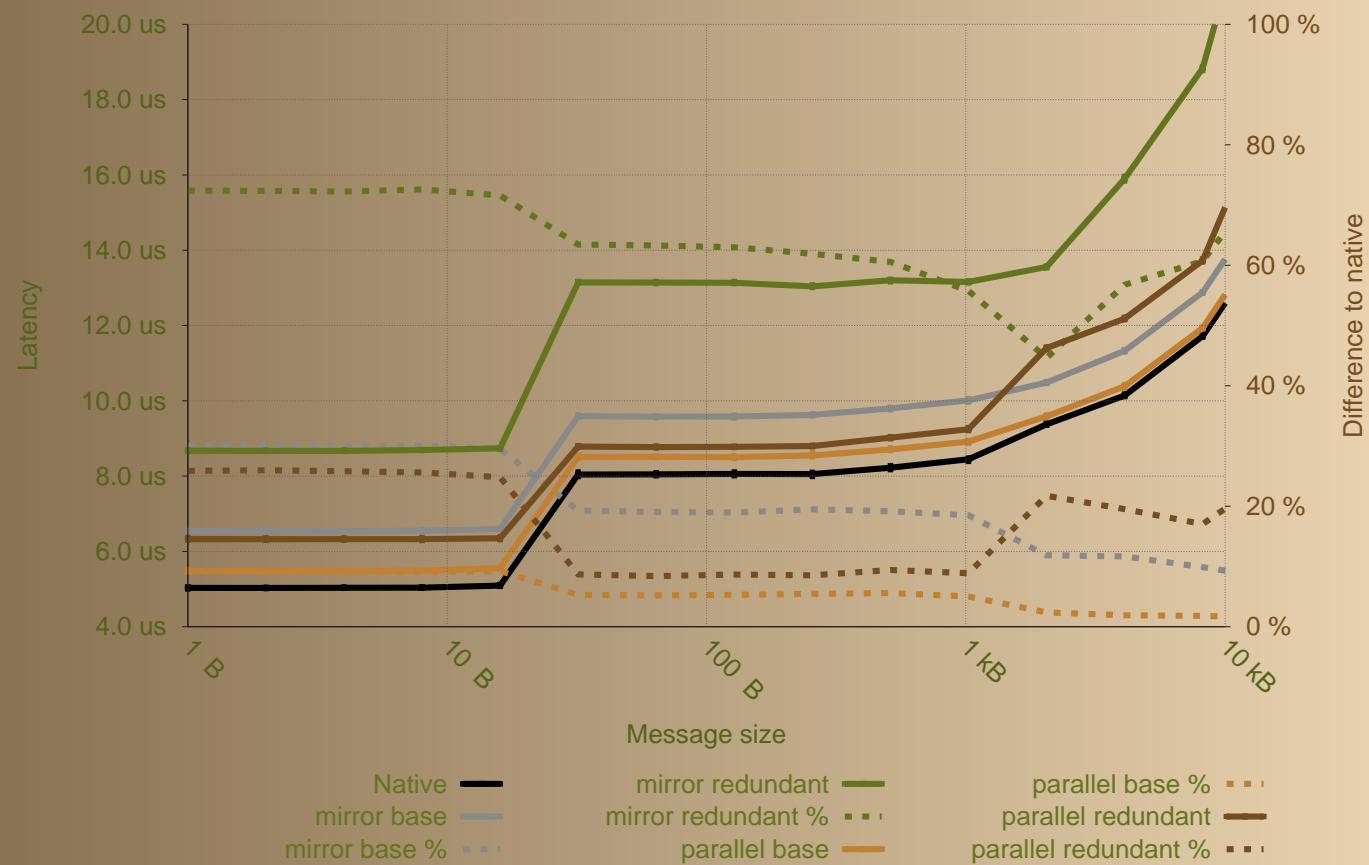
Evaluation

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



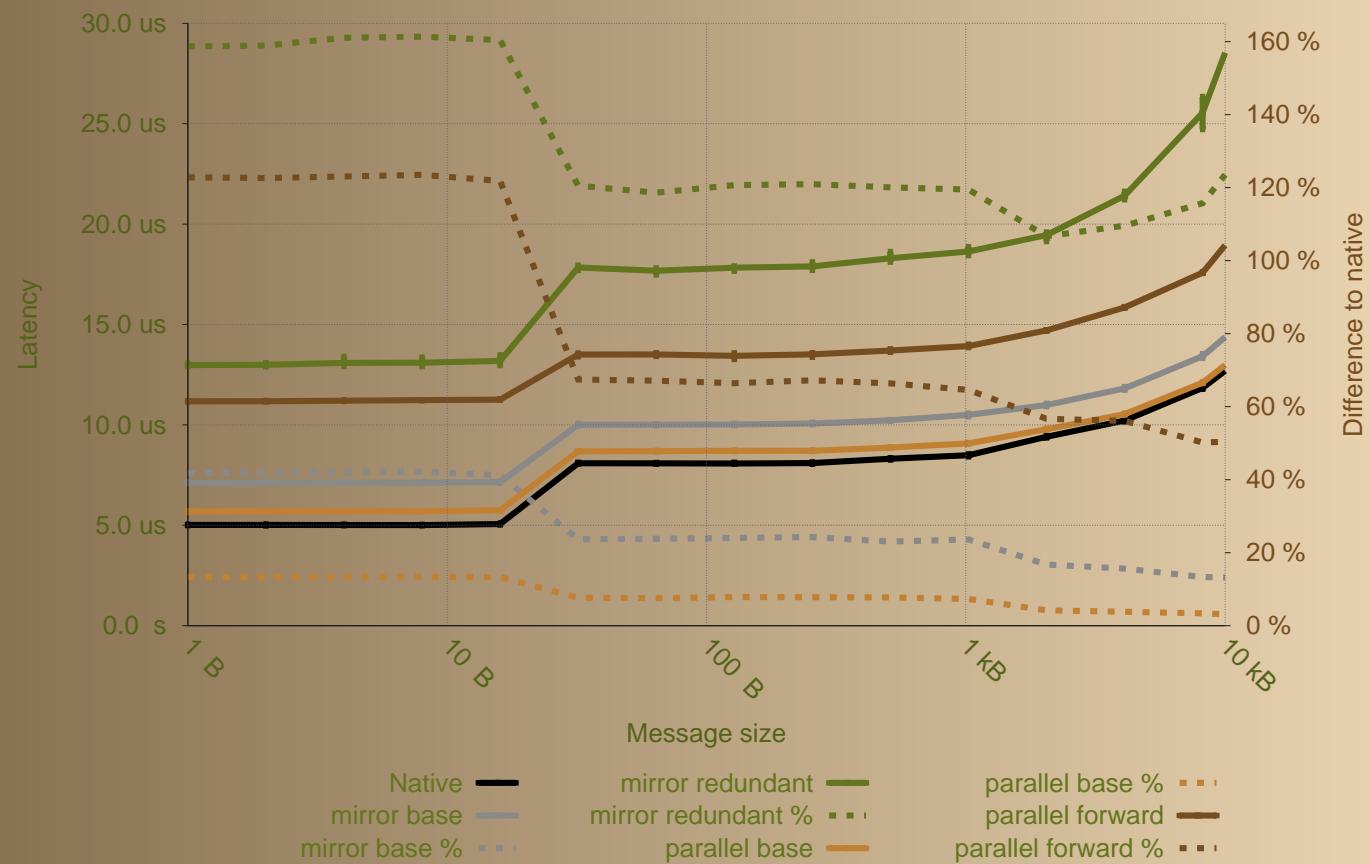
Native Benchmark w/o rMPI
Base rMPI, no redundancy

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



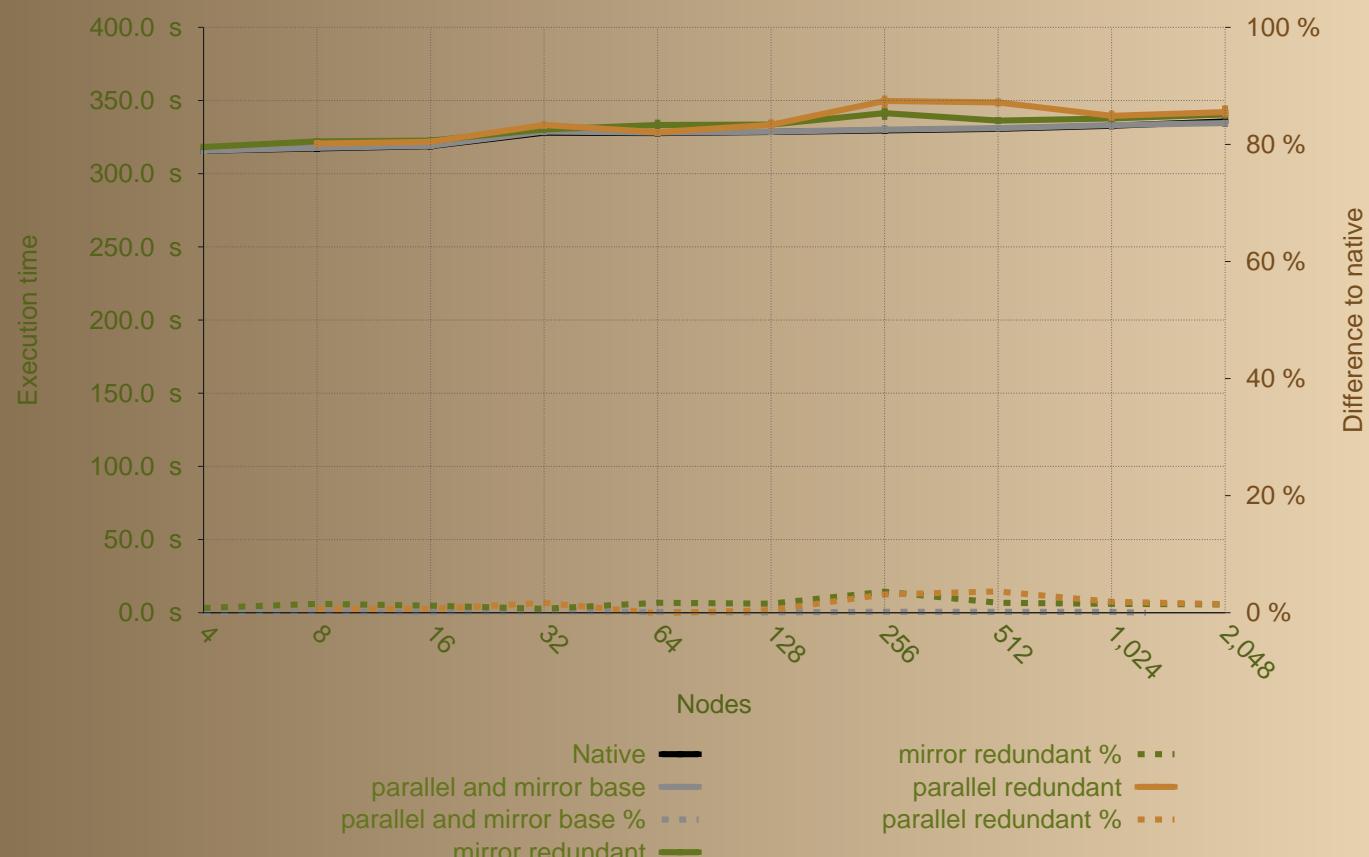
Native Benchmark w/o rMPI
Base rMPI, no redundancy

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



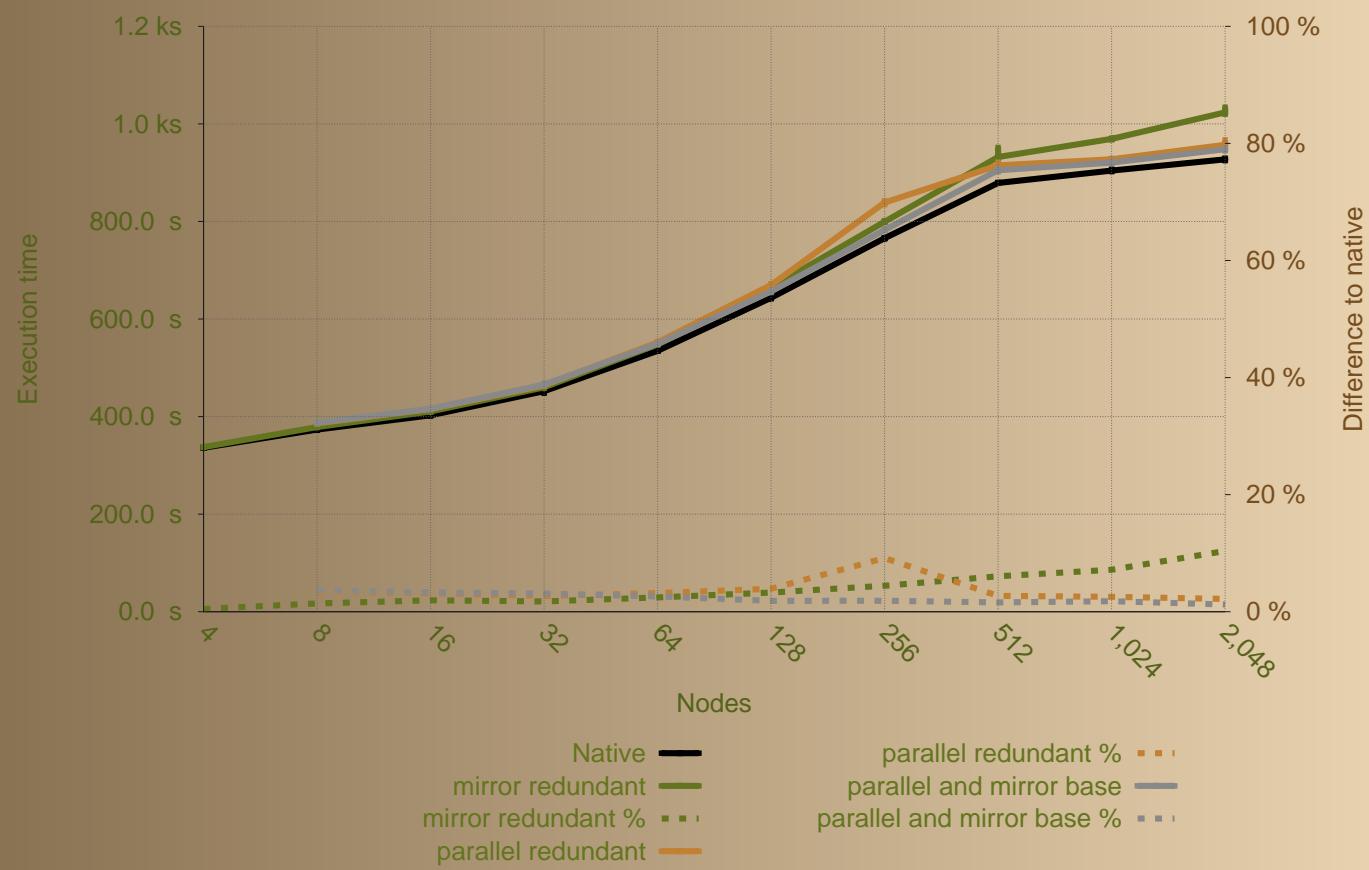
Native Benchmark w/o rMPI
Base rMPI, no redundancy

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



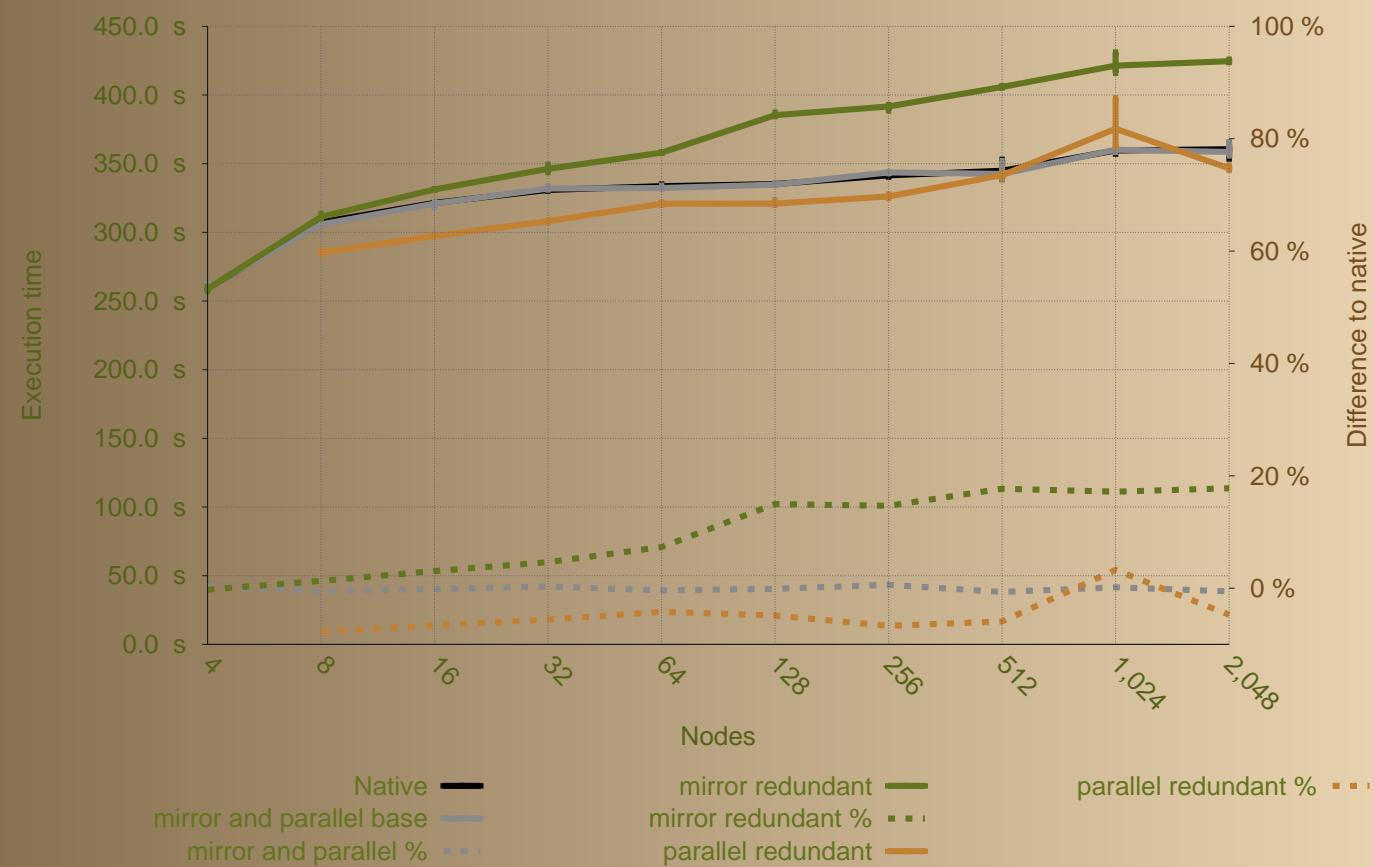
Native Benchmark w/o rMPI
Base rMPI, no redundancy

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



Native Benchmark w/o rMPI
Base rMPI, no redundancy

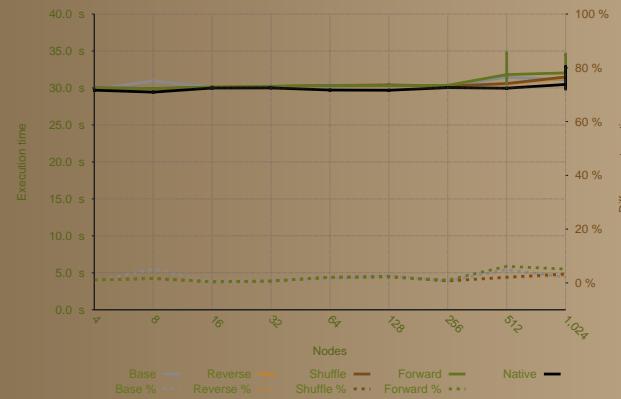
Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



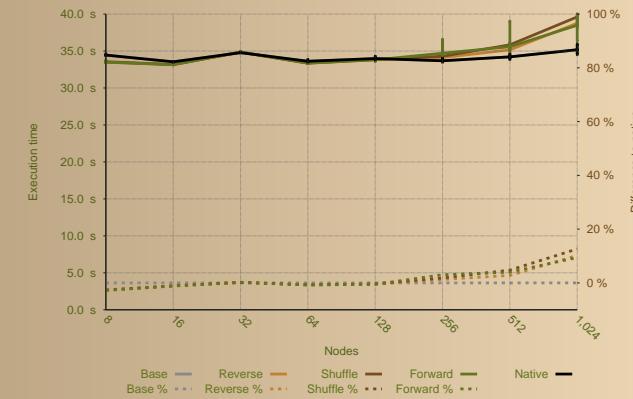
Native Benchmark w/o rMPI
Base rMPI, no redundancy

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off

Simulation
Validation
Results
Summary



(c) Mirror protocol



(d) Parallel protocol

Native Benchmark w/o *rMPI*
 Base *rMPI*, no redundancy

Motivation

Design

Evaluation

Bandwidth

Latency

LAMMPS

SAGE

CTH

HPCCG

Sys MTBF

Birthday

Trade-off

Simulation

Validation

Results

Summary

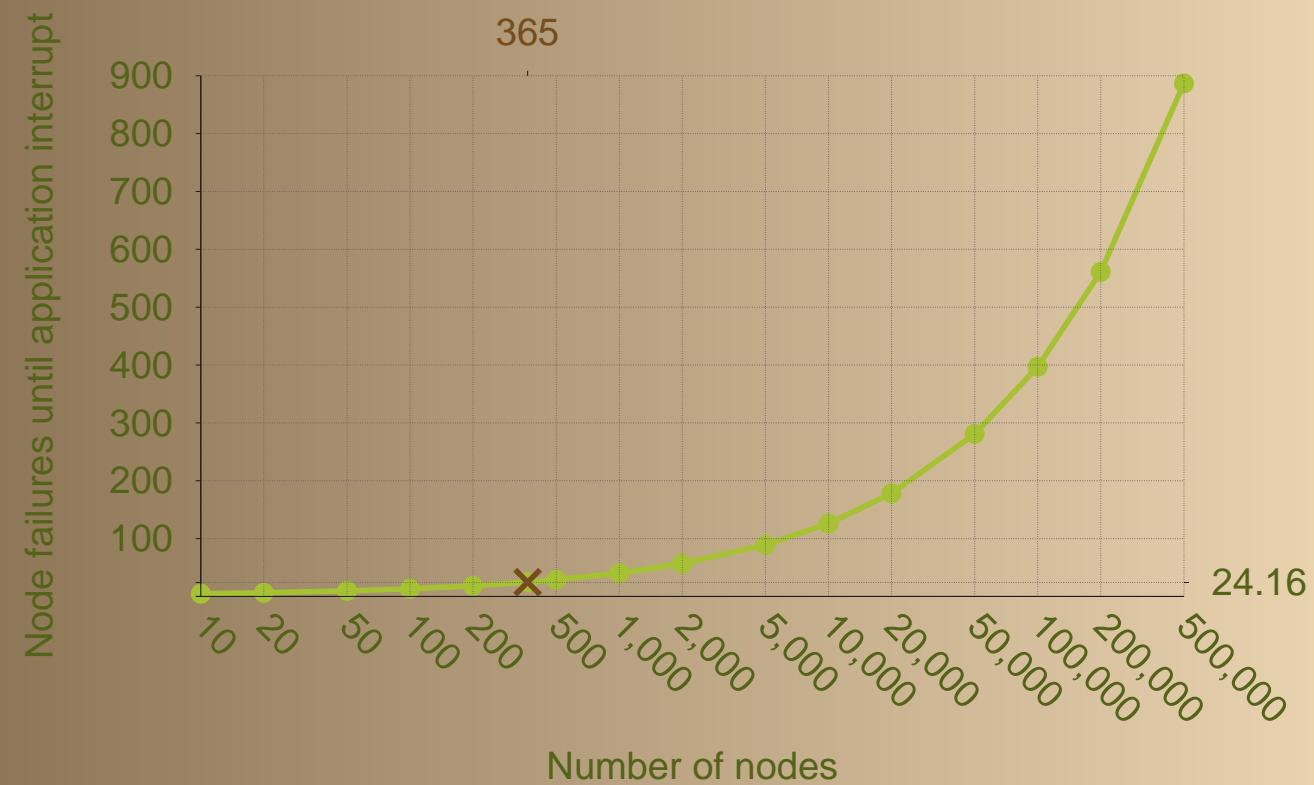
- Two nodes failing in a bundle is like two people having the same birthday

$$F(n) = 1 + \sum_{k=1}^n \frac{n!}{(n-k)! \cdot n^k} \approx \sqrt{\frac{\pi n}{2}} + \frac{2}{3} \quad (3)$$

- Using the birthday problem we can compute the system MTBF of a redundant system
- With that, we can recompute Daly's equation for redundant systems

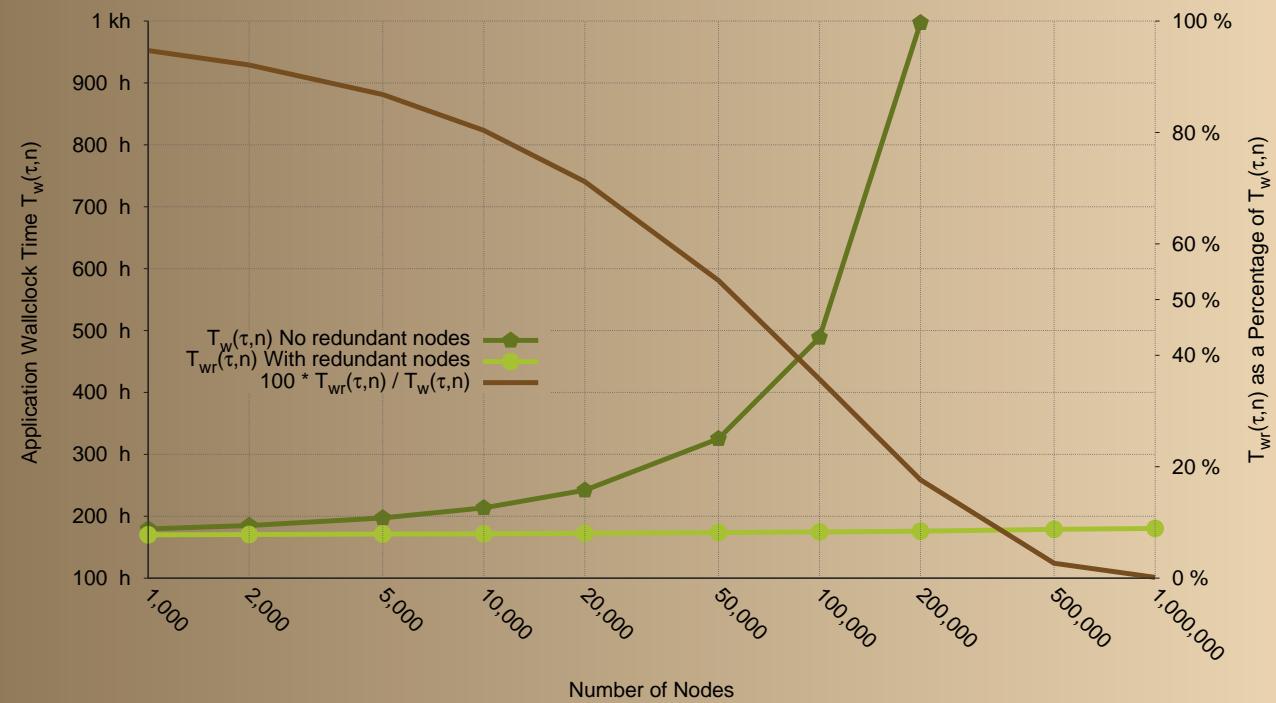
$$\Theta_{\text{app}} = \frac{\Theta_{\text{node}}}{2n} F(2n) \quad (4)$$

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



- Number of faults (interrupts) a redundant application can absorb

Motivation
Design
Evaluation
Bandwidth
Latency
LAMMPS
SAGE
CTH
HPCCG
Sys MTBF
Birthday
Trade-off
Simulation
Validation
Results
Summary



- Using Daly's equation (and ignoring overhead) we can compute when T_w is less than $1/2$ while using $2x$ nodes
- A more detailed understanding of where the time goes would be good. Maybe a simulator would help? ;-)

Motivation

Design

Evaluation

Simulation

 Motivation

 Absorb

 faults

 State diag

 Simulation

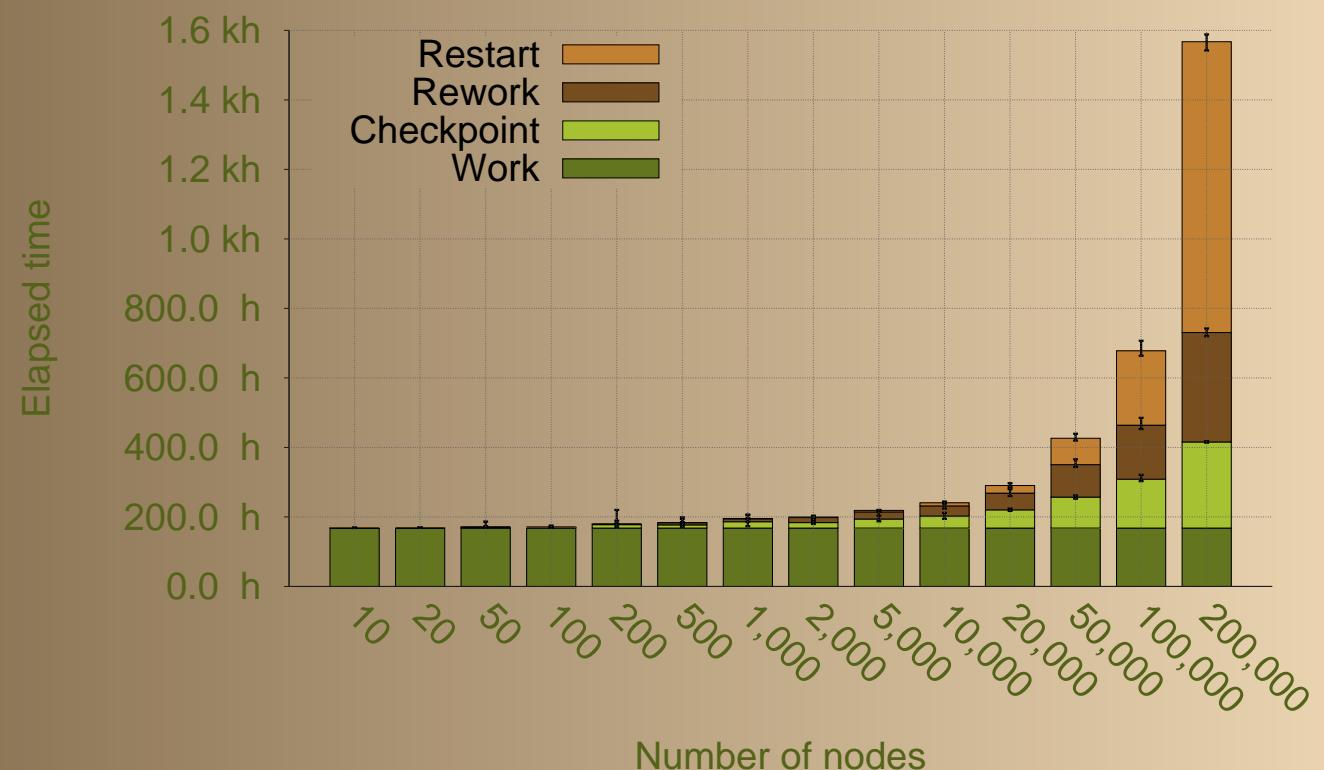
Validation

Results

Summary

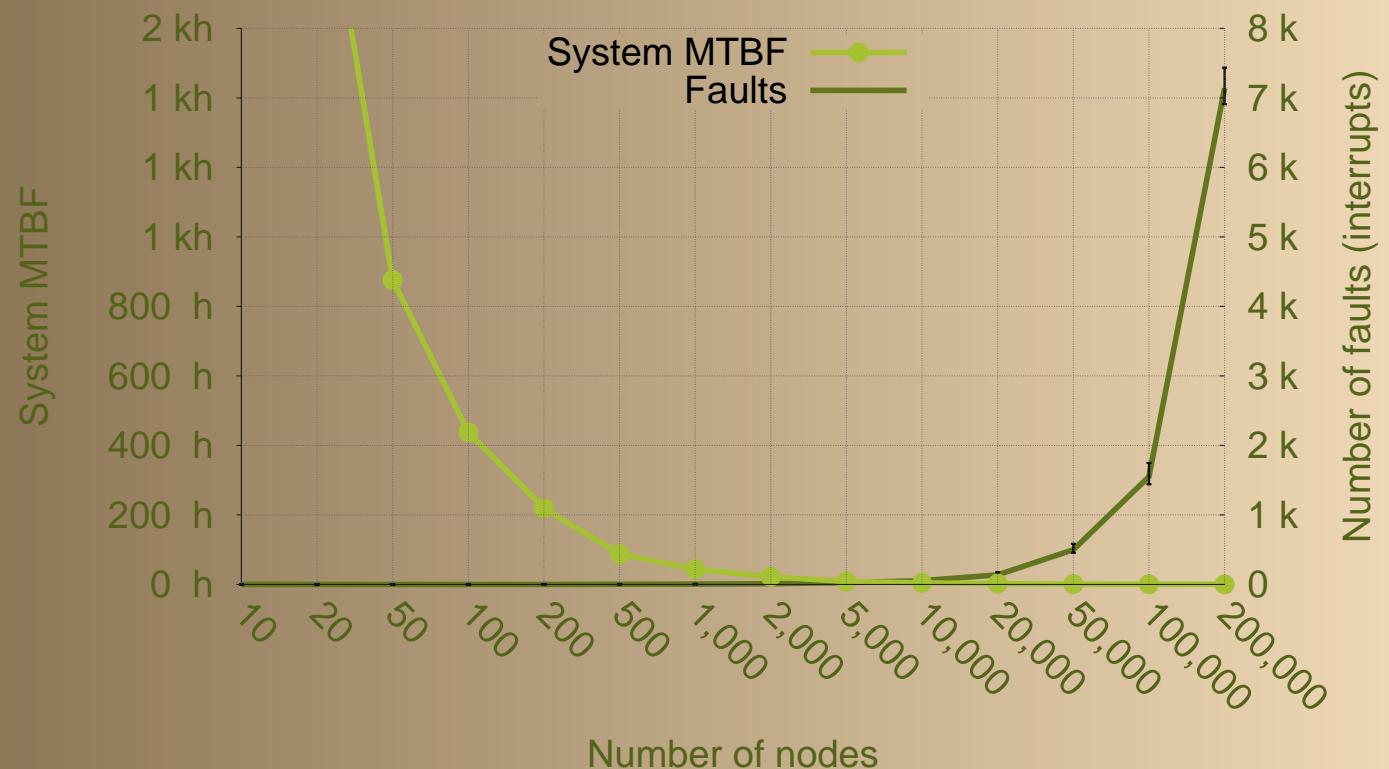
Simulation

Motivation
Design
Evaluation
Simulation
Motivation
Absorb
faults
State diag
Simulation
Validation
Results
Summary



- Simu allows detailed look and breakdown of time spent
- It also allows us to look at non-poison distribution of faults
- Other check-pointing techniques
- Consider I/O, etc.

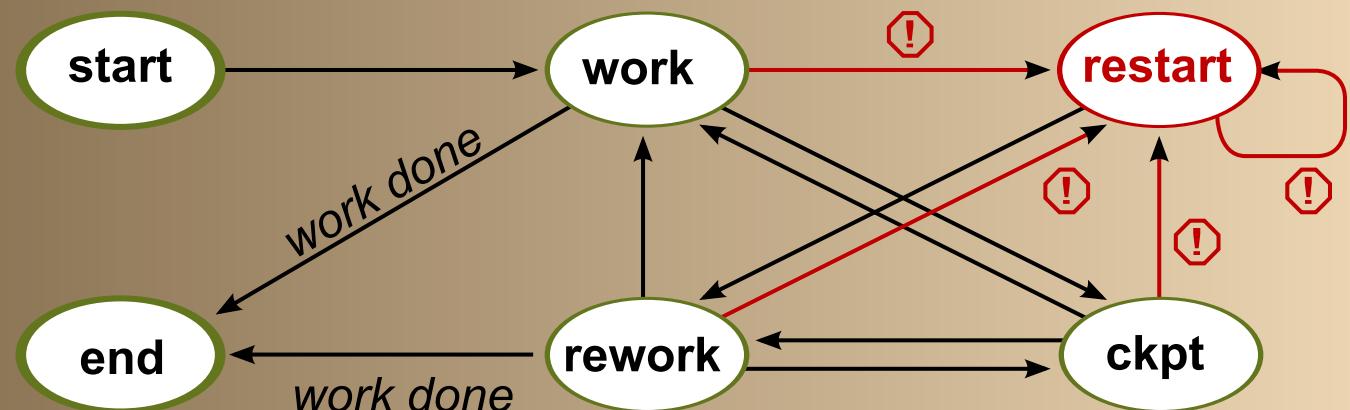
Motivation
Design
Evaluation
Simulation
Motivation
Absorb
faults
State diag
Simulation
Validation
Results
Summary



- Number of faults (interrupts) for the example on previous slide
- The left y axis shows the calculated system MTBF

Motivation
Design
Evaluation
Simulation
Motivation
Absorb
faults
State diag
Simulation

Validation
Results
Summary



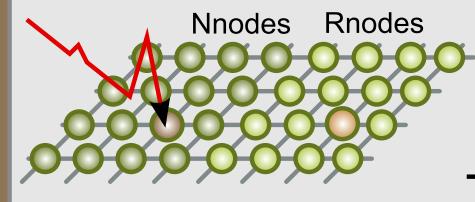
- Sim assumes coordinated check-pointing
- Other schemes may lower checkpoint time, but don't change general findings
- Work, node count, and checkpoint and restart time per simulation is fixed

Motivation
Design
Evaluation
Simulation
Motivation
Absorb
faults
State diag
Simulation
Validation
Results
Summary

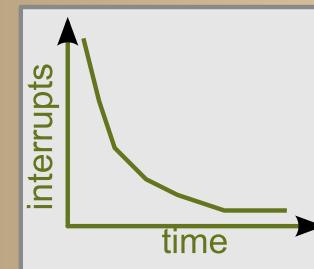
Fault generator



node
faults



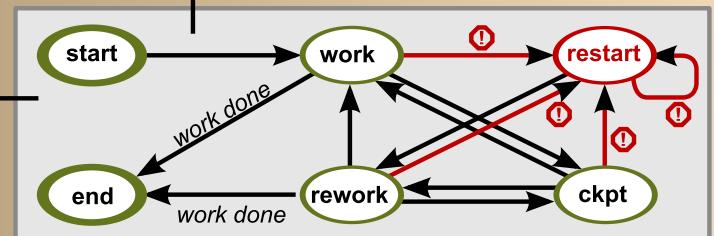
rMPI model



Analysis

request
next
interrupt
application
interrupts

statistical
data

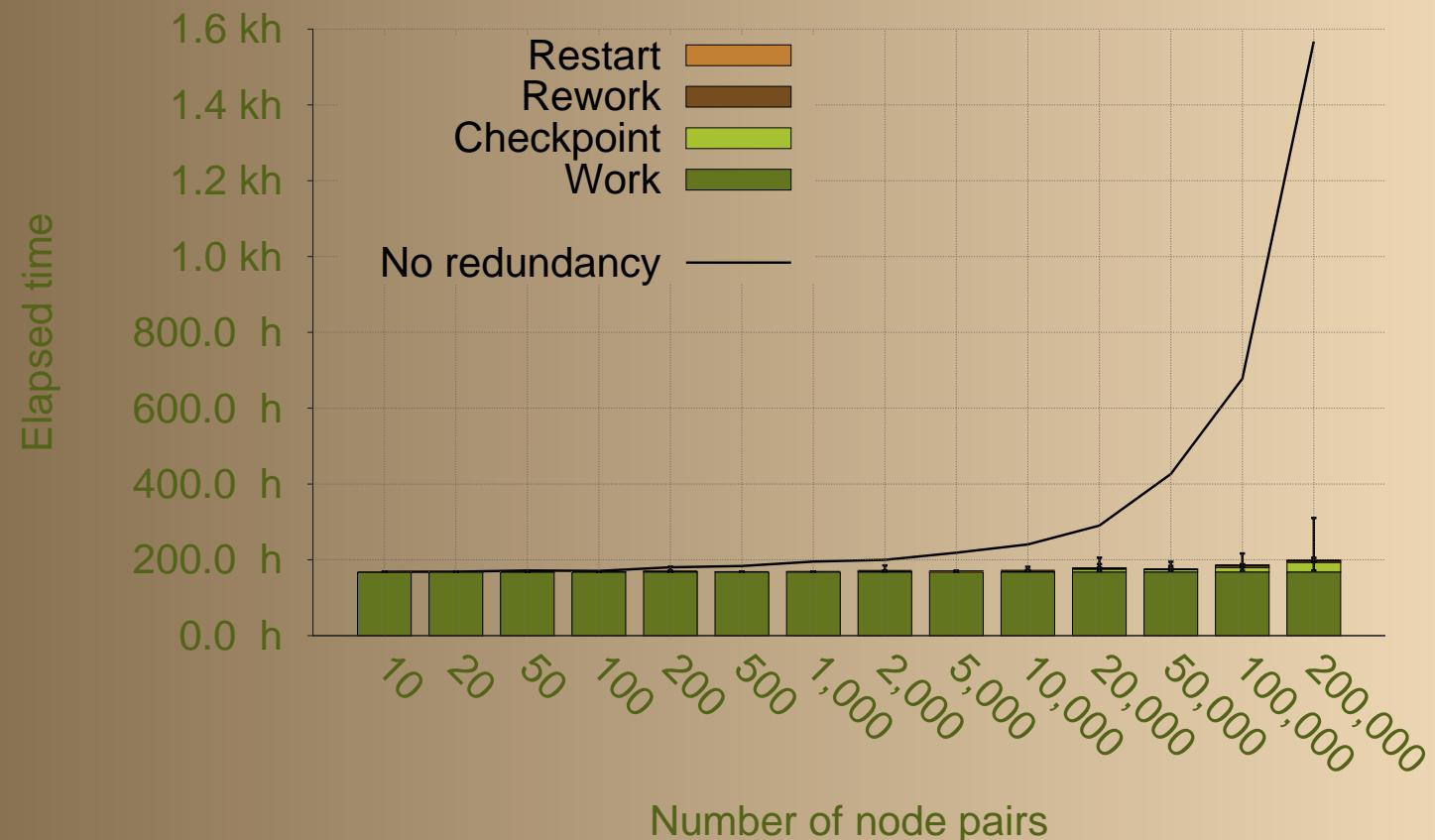


State machine

- Combine *rMPI* model with state machine
- Finish a set amount of work
- Do checkpoints and restart when necessary

Application Behavior with Redundancy

Motivation
Design
Evaluation
Simulation
Motivation
Absorb
faults
State diag
Simulation
Validation
Results
Summary



Normalized time spent in work, checkpoint, restart, and rework.
168h work, 5 min checkpoint, 10 min restart, 5-year node MTBF

Motivation

Design

Evaluation

Simulation

Validation

MTBI

Int ratio

Model

Results

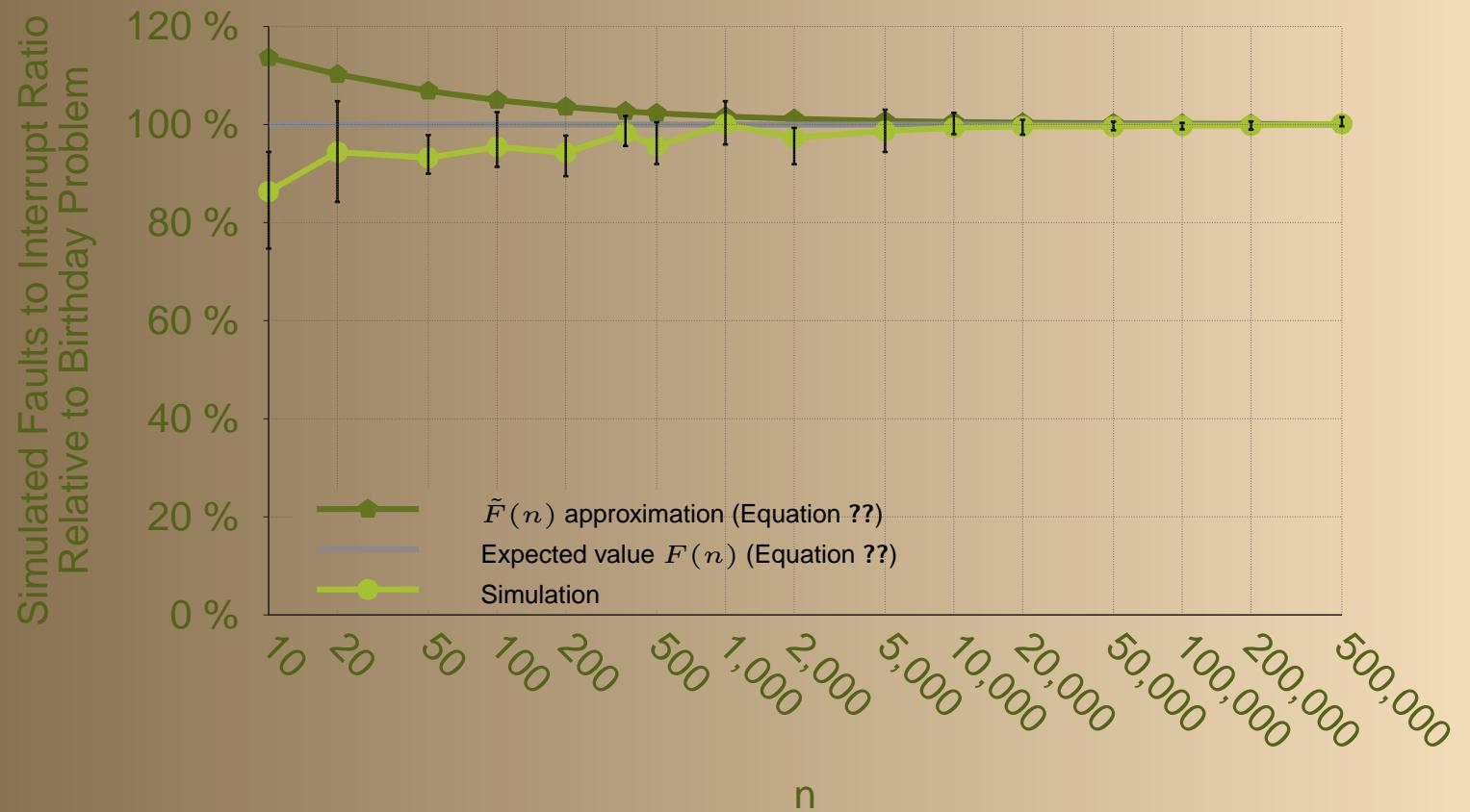
Summary

Validation

- Sim uses node MTBF as input
- Calculate mean of simulation and match it against Θ_{sys}
- Within 1% with long enough runs
- With redundant nodes, use Θ_{app} to compare (birthday problem)

Motivation
Design
Evaluation
Simulation
Validation
MTBI
Int ratio
Model
Results
Summary

- For redundant computing, ratio of faults to interrupts must match birthday problem



Motivation
Design
Evaluation
Simulation
Validation
MTBI
Int ratio
Model
Results
Summary



- Example: 5,000-hour workload on 200,000 nodes
- Counted number of faults for each interrupt
- Expected value is 561.166, simulated average was 567.093, +1%

- Compare to Daly's equation:
 - ◆ For 24-hour MTBF, sim is 5.88% higher
 - ◆ For 6-hour MTBF, sim is 11.65% higher
 - ◆ For 0.25-hour MTBF, sim is 51.7% higher
- Compare to Daly's equation with redundant nodes:
 - ◆ For 24-hour MTBF, sim is 0.28% lower
 - ◆ For 6-hour MTBF, sim is 1.27% higher
 - ◆ For 0.25-hour MTBF, sim is 3.14% higher
- Because MTBI is higher, and δ larger, due to redundant computing

Motivation

Design

Evaluation

Simulation

Validation

Results

Efficiency

App ints

Partial

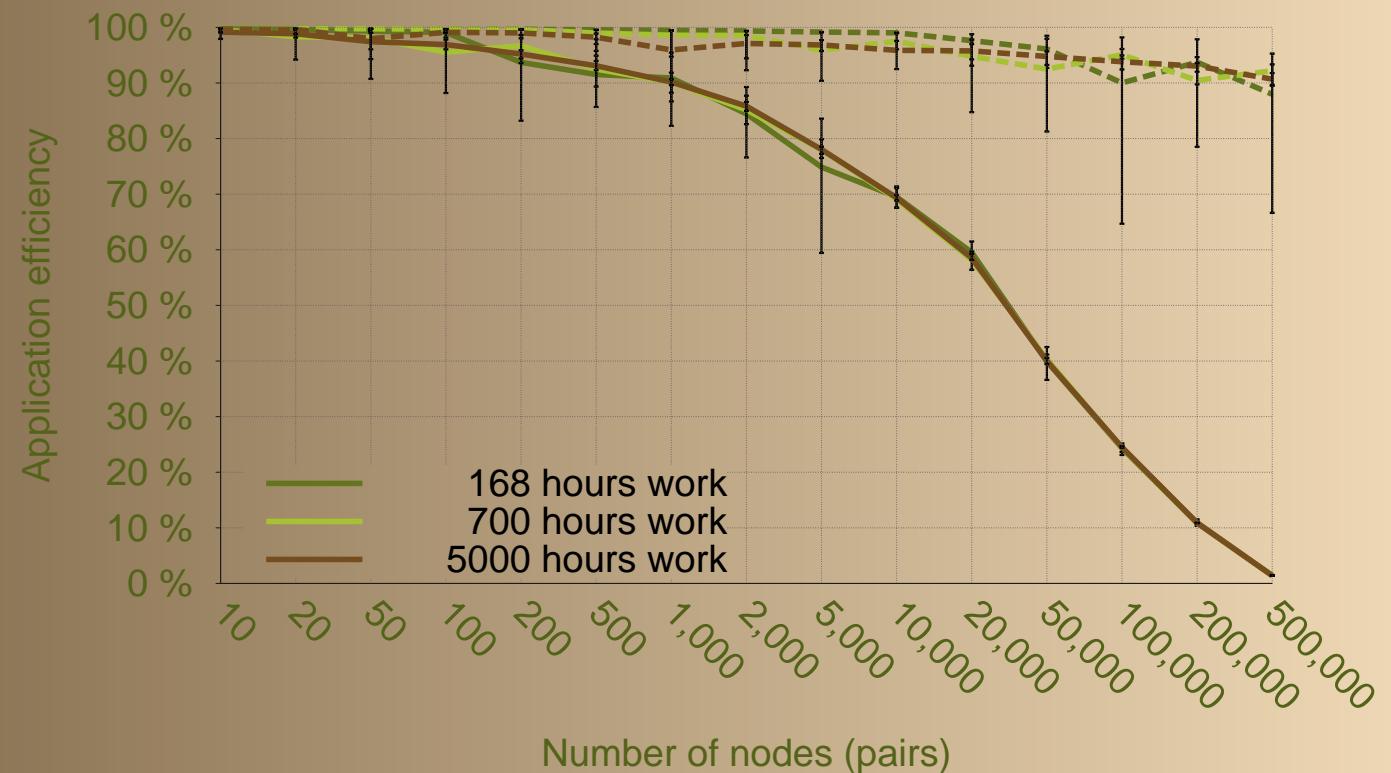
Overhead

Reboot

Summary

Results

Motivation
Design
Evaluation
Simulation
Validation
Results
Efficiency
App ints
Partial
Overhead
Reboot
Summary

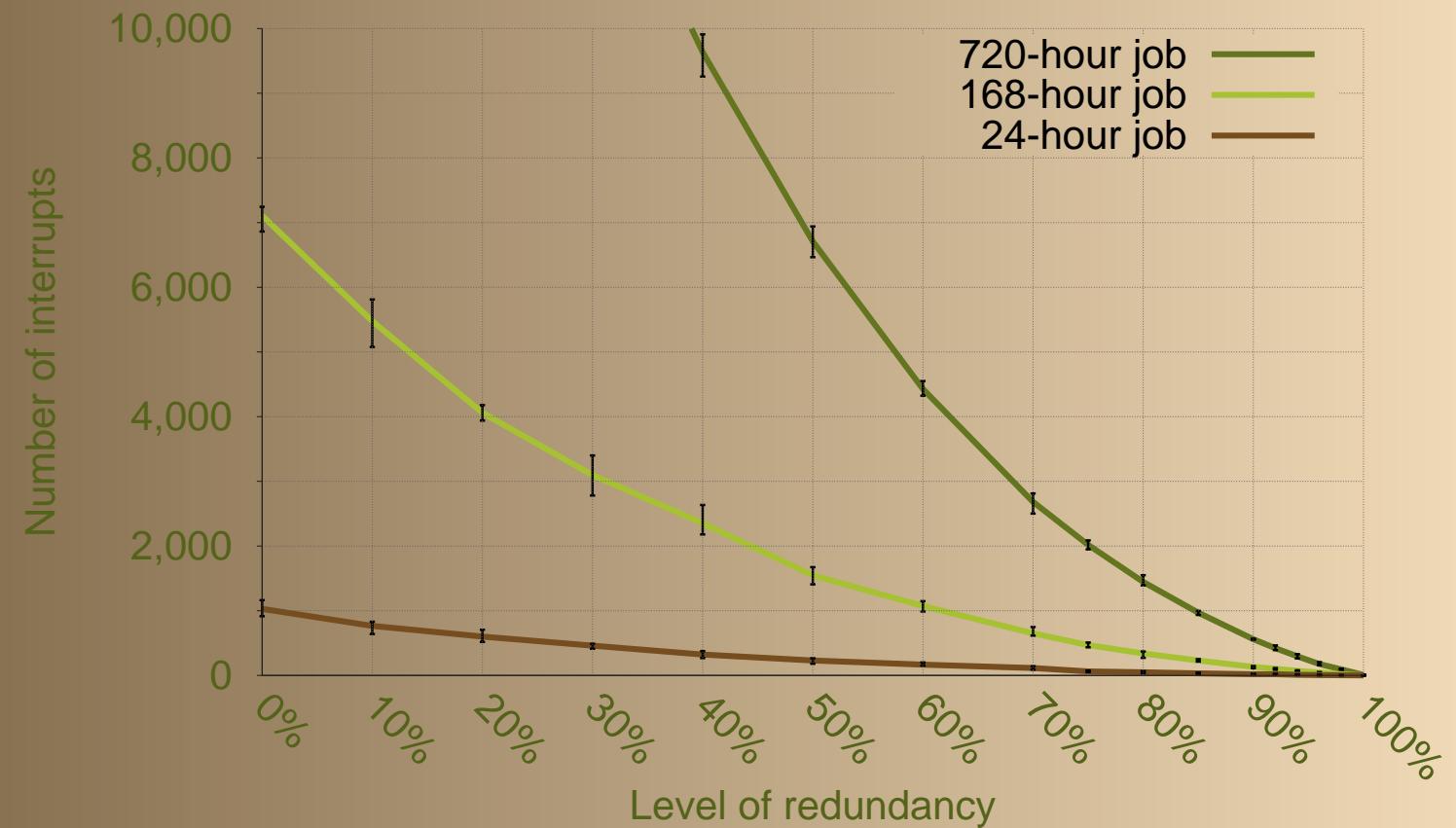


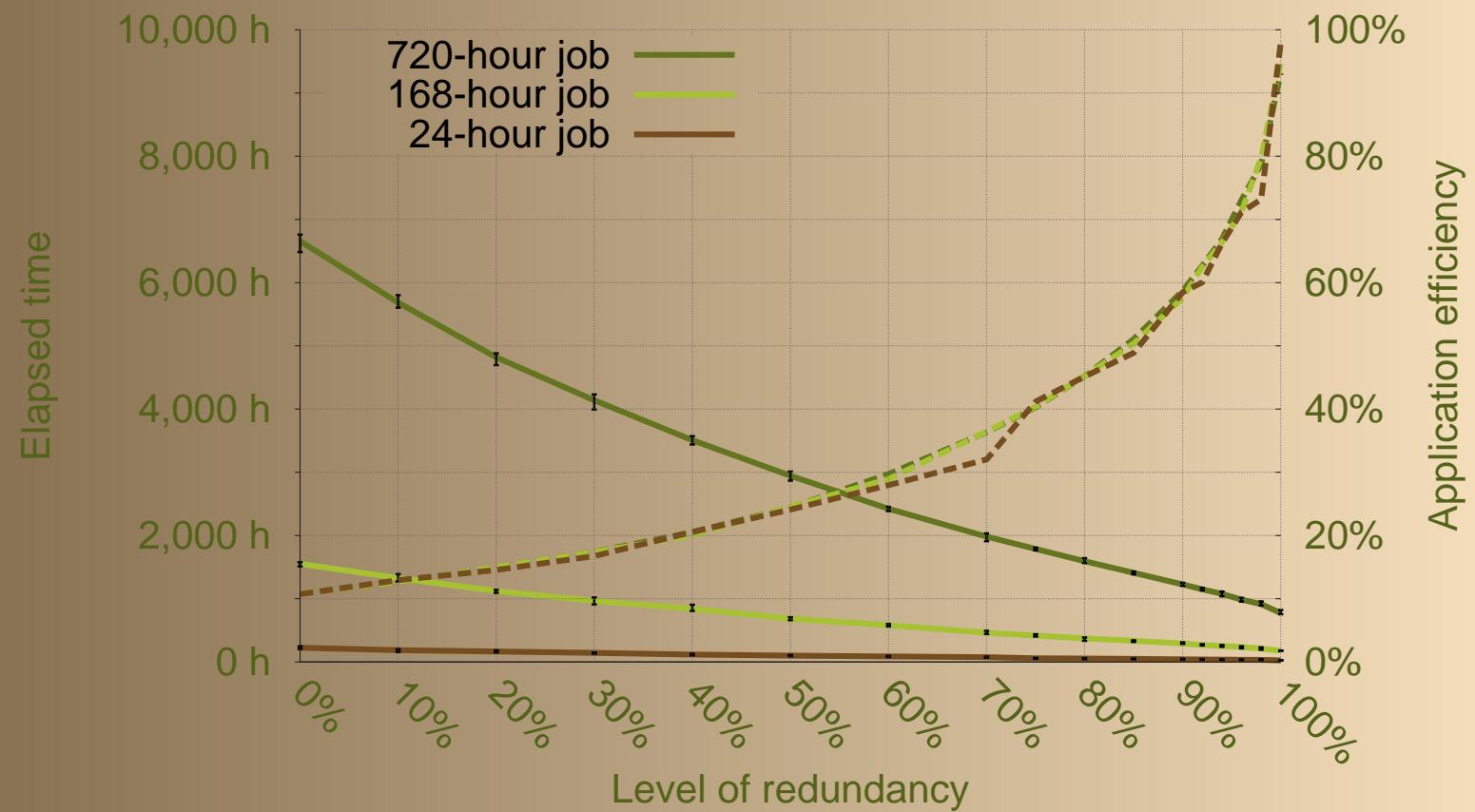
- Example for 168 hours (a week), 700 hours (a month), and 5,000 hours (seven months)
- Application efficiency impacts system throughput

- Longer time on system means more faults

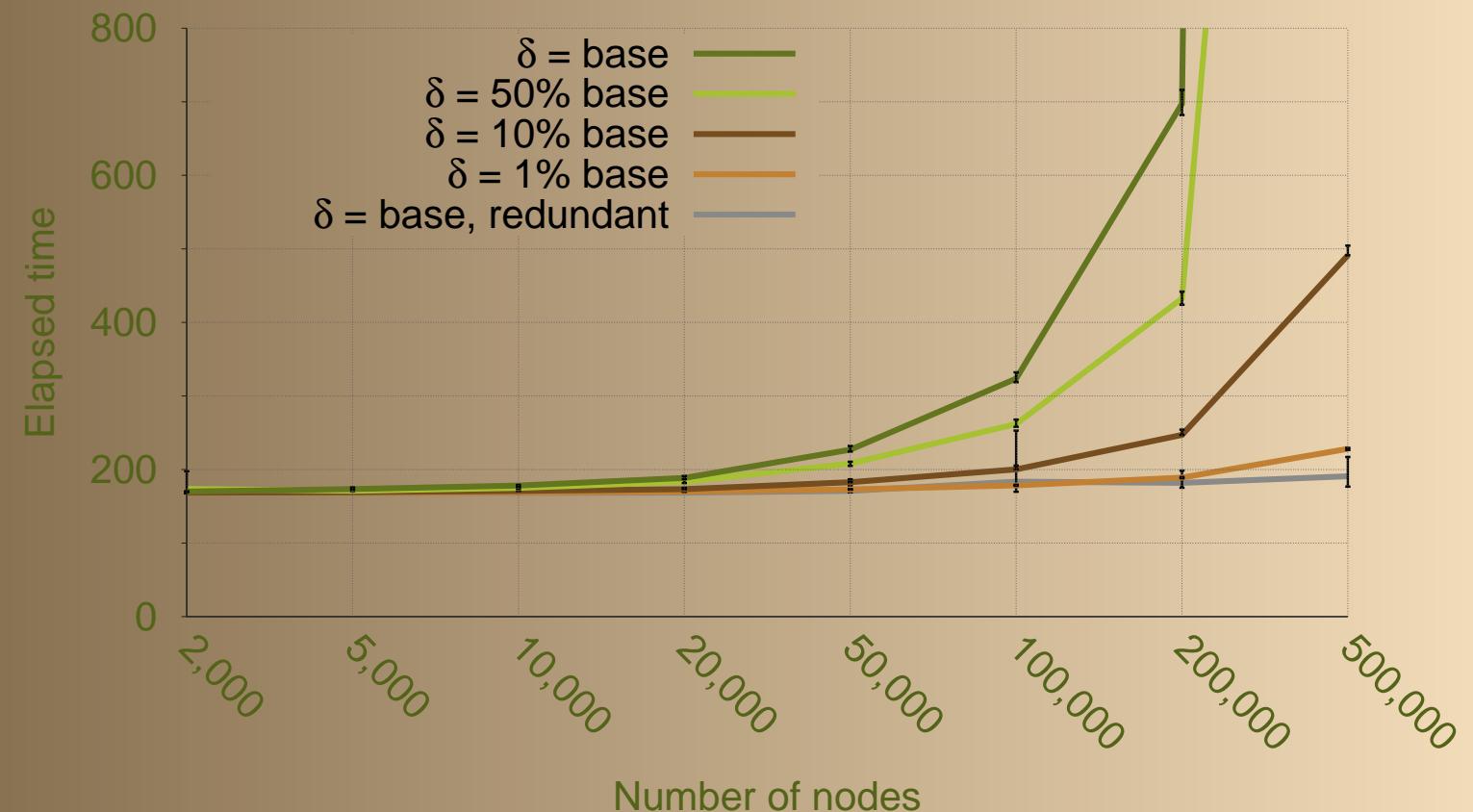
	nodes (pairs)	not redundant		redundant	
		faults	interrupts	faults	interrupts
Results	100	11	11	24	1
Efficiency	500	60	60	117	3
App ints	1,000	126	126	231	3
Partial Overhead	2,000	264	264	447	5
Reboot	5,000	739	739	1,153	9
Summary	10,000	1,653	1,653	2,384	13
	20,000	3,902	3,902	4,784	19
	50,000	14,228	14,228	11,973	30
	100,000	46,565	46,565	24,304	43
	200,000	209,909	209,909	48,930	62
	500,000	4,031,114	4,031,114	125,811	101

Motivation
Design
Evaluation
Simulation
Validation
Results
Efficiency
App ints
Partial
Overhead
Reboot
Summary

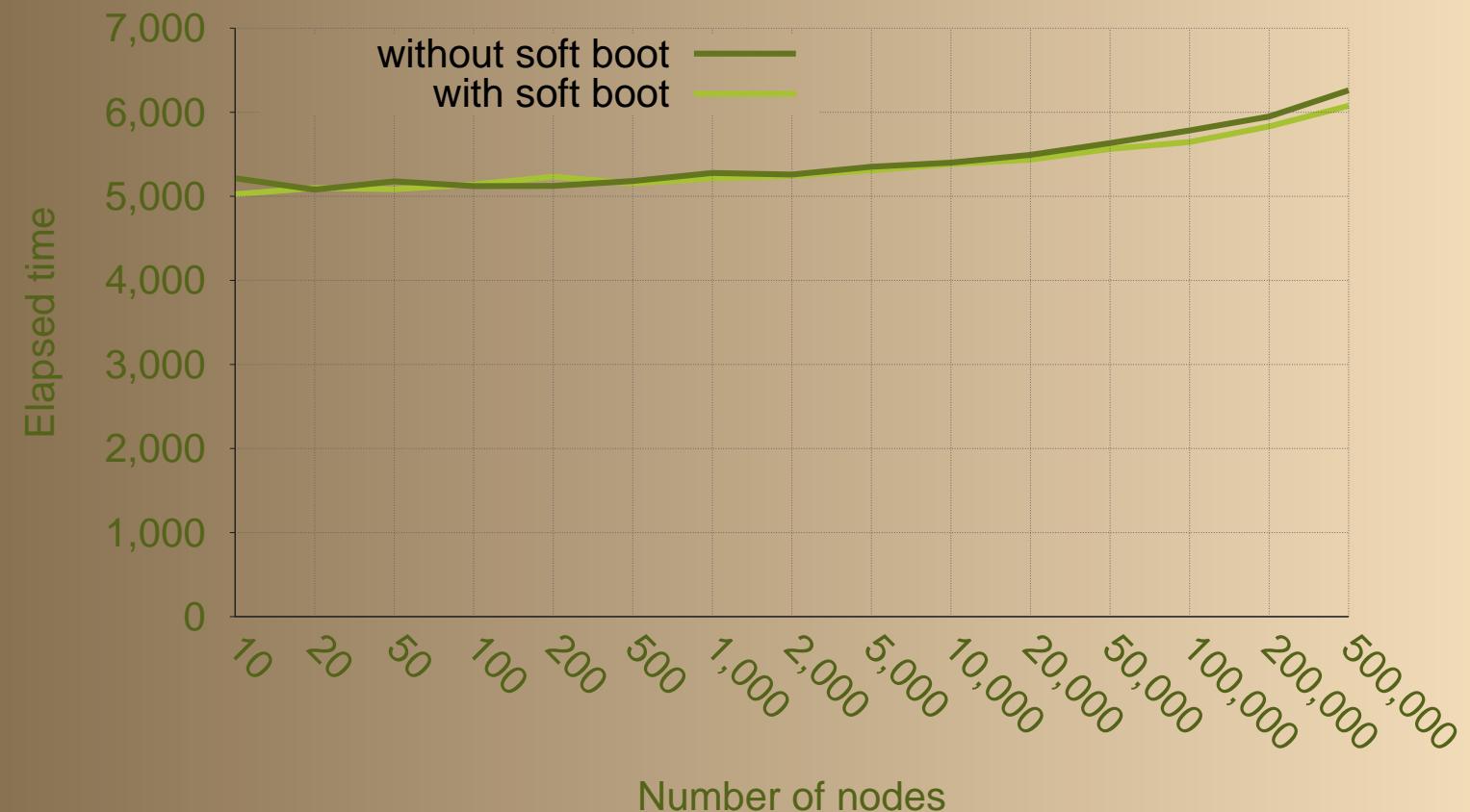




Motivation
Design
Evaluation
Simulation
Validation
Results
Efficiency
App ints
Partial
Overhead
Reboot
Summary



Motivation
Design
Evaluation
Simulation
Validation
Results
Efficiency
App ints
Partial
Overhead
Reboot
Summary



Motivation
Design
Evaluation
Simulation
Validation
Results
Summary
People

Summary

- Redundant MPI library can be done at user level
- Overhead for applications is not significant for most applications
- Application restart simulator allows modeling of
 - ◆ varying node counts
 - ◆ node MTBF
 - ◆ level of redundancy
 - ◆ failure distribution function (exponential, gamma, weibull)
 - ◆ amount of work, checkpoint and restart times
- Model helps determine when redundancy pays off

- Kurt Ferreira
- Ron Oldfield
- Jon Stearley
- James Laros
- Kevin Pedretti
- Ron Brightwell